IN THE CLAIMS

Please amend the claims as follows:

1. (Currently Amended) A method ~~for modifying a software application~~ comprising:

~~inserting function calls at entry points and exit points of each method associated with the software application via a bytecode modifier;~~

modifying a classfile after said classfile has been compiled from a source code version of the software application, said classfile describing properties of a class within an object oriented environment[[;]], said modifying comprising

modifying a method information structure for each method associated with the software application by adding byte code instructions to said method information structure to cause a plug-in handler method associated with a plug-in handler to execute an output function for each method, the plug-in handler to record method information associated with methods at each entry point and exit point, modification of the method information structure for each method comprising inserting function calls at entry points and exit points of each method associated with the software application via a bytecode modifier,[[;]]

the method further comprising:

compiling results of the ~~modifying of the~~ modified classfile and providing the results to a graphical user interface to create a graphical representation of the results, the results including method information, the method information including a dependency hierarchical tree indicating dependency order of the methods, and a time hierarchical tree indicting chronological order of the methods; and

filtering the method information, via a filtering module, according to user preferences and the dependency and time hierarchical trees.

2. (Canceled)

3. (Currently Amended) The method of claim [[2]] 1 further comprising:

adding a field information structure to the methods, said field information structure describing a field that is to store a numeric identifier of said class.

4. (Previously Presented) The method of claim 3 wherein said numeric identifier is provided to said class by a method of which a dispatch unit is comprised.

5. (Previously Presented) The method of claim 1 wherein a portion of said byte code instructions that are added to said method are for causing said plug-in handler method to provide an output function treatment in response to an entry point of said method being reached.

6. (Previously Presented) The method of claim 5 wherein said output function treatment is a function selected from the group consisting of:

1) recording a time of entry for said method;
2) recording an input parameter value for said method; and,
3) incrementing a counter for said method.

7. (Previously Presented) The method of claim 1 wherein a portion of said byte code instructions that are added to said method are for causing said plug-in handler method to provide said output function treatment in response to an exit point of said method being inevitably reached.

8. (Previously Presented) The method of claim 7 wherein said output function treatment is a function selected from the group consisting of:

1) recording a time of entry for said method;
2) recording an input parameter value for said method; and
3) incrementing a counter for said method.

9. (Previously Presented) The method of claim 7 wherein the portions of said byte code instructions that are added to said method are for causing said plug-in handler method to provide

said output function treatment in response to any exit point of said method being inevitably reached.

10. (Previously Presented) The method of claim 1 wherein a portion of said byte code instructions that are added to said method are for causing said plug-in molecule's handler method to provide said output function treatment in response to an error arising during execution of said method.

11-12 (Canceled)

13. (Previously Presented) The method of claim 1 wherein said byte code instructions are Java compatible and wherein said at least one of said instructions is an invokestatic instruction.

14. (Previously Presented) The method of claim 1 wherein said byte code instructions are Java compatible and wherein said at least one of said instructions is an invokevirtual instruction.

15. (Previously Presented) The method of claim 1 wherein said byte code instructions are Java compatible and wherein said at least one of said instructions is an invokespecial instruction.

16-21 (Canceled)

22. (Currently Amended) A machine readable storage medium comprising instructions ~~to modify a software application~~ which, when executed by a machine, cause a machine to:

~~inserting functions calls at entry points and exit points of each method associated with the software application via a bytecode modifier;~~

~~modifying~~ modify a classfile after said classfile has been compiled from a source code version of the software application, said classfile describing properties of a class within an object oriented environment[[;]], said modifying comprising:

~~modify~~ modifying a method information structure for each method associated with the software application by adding byte code instructions to said method information

structure to cause a plug-in handler method associated with a plug-in handler to execute an output function for each method, the plug-in handler to record method information associated with methods at each entry point and exit point, modification of the method information structure for each method comprising inserting function calls at entry points and exit points of each method associated with the software application via a bytecode modifier,[[;]]

and further to:

compile results of the ~~modifying of the~~ modified classfile and provide the results to a graphical user interface to create a graphical representation of the results, the results including method information, the method information including a dependency hierarchical tree indicating dependency order of the methods, and a time hierarchical tree indicating chronological order of the methods; and

filter the method information, via a filtering module, according to user preferences and the dependency and time hierarchical trees.

23. (Canceled)

24. (Currently Amended) The machine readable storage medium of claim [[23]] 22 wherein the instructions which, when executed, further cause the machine to:

add a field information structure to the methods, said field information structure describing a field that is to store a numeric identifier of said class.

25. (Canceled)

26. (Previously Presented) The machine readable storage medium of claim 22 wherein a portion of said byte code instructions that are added to said method are for causing said plug-in handler method to provide said output function treatment in response to an entry point of said method being reached.

27. (Canceled)

28. (Previously Presented) The machine readable storage medium of claim 22 wherein a portion of said byte code instructions that are added to said method are for causing said plug-in handler method to provide said output function treatment in response to an exit point of said method being inevitably reached.

29. (Canceled)

30. (Previously Presented) The machine readable storage medium of claim 28 wherein the portions of said byte code instructions that are added to said method are for causing said plug-in handler method to provide said output function treatment in response to any exit point of said method being inevitably reached.

31. (Previously Presented) The machine readable storage medium of claim 22 wherein a portion of said byte code instructions that are added to said method are for causing said plug-in handler method to provide said output function treatment in response to an error arising during execution of said method.

32-42. (Canceled)

43. (Currently Amended) A system ~~for modifying a software application~~ comprising:

~~means for inserting functions calls at entry points and exit points of each method associated with the software application via a bytecode modifier;~~

means for modifying a classfile after said classfile has been compiled from a source code version of the software application, said classfile describing properties of a class within an object oriented environment[[;]], said modifying comprising:

~~means for~~ modifying a method information structure for each method associated with the software application by adding byte code instructions to the said method information structure to cause a plug-in handler method associated with a plug-in handler

to execute an output function for said each method, the plug-in handler to record method information associated with methods at each entry point and exit point, modification of the method information structure for each method comprising inserting function calls at entry points and exit points of each method associated with the software application via a bytecode modifier,[[;]]

the system further comprising:

means for compiling results of the ~~modifying of the~~ modified classfile and for providing the results to a graphical user interface to create a graphical representation of the results, the results including method information, the method information including a dependency hierarchical tree indicating dependency order of the methods, and a time hierarchical tree indicting chronological order of the methods; and

means for filtering the method information, via a filtering module, according to user preferences and the dependency and time hierarchical trees.

44. (Previously Presented) The system of claim 43 wherein said identities are each in a character string format.

45. (Currently Amended) The system of claim [[44]] 43 wherein further comprising:

means for adding a field information structure to the methods, said field information structure describing a field that is to store a numeric identifier of said class.

46. (Canceled)

47. (Previously Presented) The system of claim 43 wherein a portion of said byte code instructions that are added to said method are causing said plug-in handler method to provide said output function treatment in response to an entry point of said method being reached.